A general model for performance optimization of sequential systems

Dmitry Bufistov
Univ. Politècnica de Catalunya
Barcelona, Spain

Jordi Cortadella
Univ. Politècnica de Catalunya
Barcelona, Spain

Mike Kishinevsky
Strategic CAD Lab, Intel Corp.
Hillsboro, OR, USA

Sachin Sapatnekar
University of Minnesota
Minneapolis, MN 55455

***Abstract*— Retiming, c-slow retiming and recycling are different transformations for the performance optimization of sequential circuits. For retiming and c-slow retiming, different models that provide exact solutions have already been proposed. An exact model for recycling was yet unknown. This paper presents a general formulation that covers the combination of the three schemes for performance optimization. It provides an exact model based on integer linear programming that resorts to the structural theory of marked graphs. A set of experiments has been designed to show the benefits in performance obtained by combining retiming and recycling. The results also show the applicability of the method in large circuits.**

I. Introduction

A. Retiming

Retiming [1] has been the traditional technique for sequential optimization. Retiming can explore different configurations of sequential circuits by moving flip-flops across combinational logic without changing their functionality. In this way, solutions trading-off area, delay and power can be explored using this flexibility. Different approaches have been used to efficiently solve retiming [2]. The analogy between the skew optimization and the retiming problems has also been exploited to propose efficient algorithms for large circuits [3].

As interconnect delays become more dominant, some global wires may have long delays that degrade the overall system performance. Under these conditions, retiming can be combined with wire pipelining [4]. However, since the number of flip-flops between inputs and outputs must remain unchanged, retiming is not always capable of meeting the intended clock period.

B. C-slow retiming

C-slow retiming [1] was proposed as a technique to accommodate the frequency of a circuit to its environment at the expense of reducing the input issue rate, i.e., number of cycles between consecutive input data. By c-slowing a circuit, the throughput (number of input data processed per cycle) is reduced to $1/c$. The main advantage of c-slow retiming is the low control overhead associated to the enabling of the flip-flops. However, extra registers must be inserted to equalize the input rate of all the cycles in the circuit. This often results in a tangible area overhead. A method to find the optimal value for c and minimize the extra flip-flops added to the circuit was presented in [5].

C. Recycling

Latency insensitivity (LI) [6] was proposed as a paradigm to design systems that are tolerant to the variability of communication and computation delays. A common strategy to design LI systems is to include a wrapper around each component that encapsulates its functionality. The wrapper contains buffers that synchronize the inputs and propagate the outputs in such a way that the functionality of the system is preserved regardless the computation and communication delays of its components.

LI design enables the automatic insertion of *empty buffers* to improve the performance of the system. The transformation for inserting empty buffers (also known as *relay stations* in LI terminology) is called *recycling* [7]. Intuitively, in LI systems every data item has an associated *valid* bit. When the valid bit associated to a register is not asserted, the data stored in the register is assumed to be non-informative.

LI design can be viewed as a discretized asynchronous design. The concept of inserting empty buffers for optimizing system performance were long known in the asynchronous design under different names of *bubble insertion* and *slack matching* [8], [9]. In [10], [11], exact algorithms for slack matching on choice-free asynchronous systems were presented. This problem is similar to solving the recycling problem in isolation. These papers however did not deal with the combined retiming and recycling problem.

Recycling requires more complex control but provides an additional degree of flexibility for adapting clock period and throughput to the requirements of the environment. Unlike c-slow, recycling does not require all cycles to be equalized to the same input rate. However, the control must explicitly take care of the back-pressure when not all inputs are available at one of the computational blocks of the circuit. For this reason, a FIFO-based communication using a handshake protocol is required to handle this flexibility.

However, some recent techniques for LI design have been able to propose schemes that require very low extra overhead. In particular, by using a static scheduling, the logic for the handshake protocol and the area occupied by the sequential elements can be drastically reduced [12]. The scheme presented in [13] also proposed a latch-based implementation for the storage elements that make the datapath overhead negligible with regard to a conventional implementation with flip-flops.

D. Retiming and recycling

Retiming and recycling (R&R) can join their efforts to find configurations with better performance than the ones that would be obtained by using only each one of them individually [14].

Figure 1 shows an example on how retiming and recycling can collaborate. The nodes (circles) of the figure are labeled with their delays. The boxes (holding a dot) represent the registers. Figure 1(a) depicts an optimal retiming of the circuit, with a cycle time of 16 time units.

Figure 1(b) shows a configuration obtained by 2-slowing the circuit. The empty boxes represent registers with non-valid data. Henceforth, we will distinguish between *dots* and *bubbles* to denote registers with valid and non-valid data, respectively.

The 2-slow solution can achieve a cycle of 10 time units. However, the *throughput* is degraded to 1/2, i.e., one data item is processed every two cycles. This gives an *effective cycle time* of 20. It means that a dot is processed every 20 time units, compared to the 16 time

1-4244-1382-6/07/$25.00 ©2007 IEEE

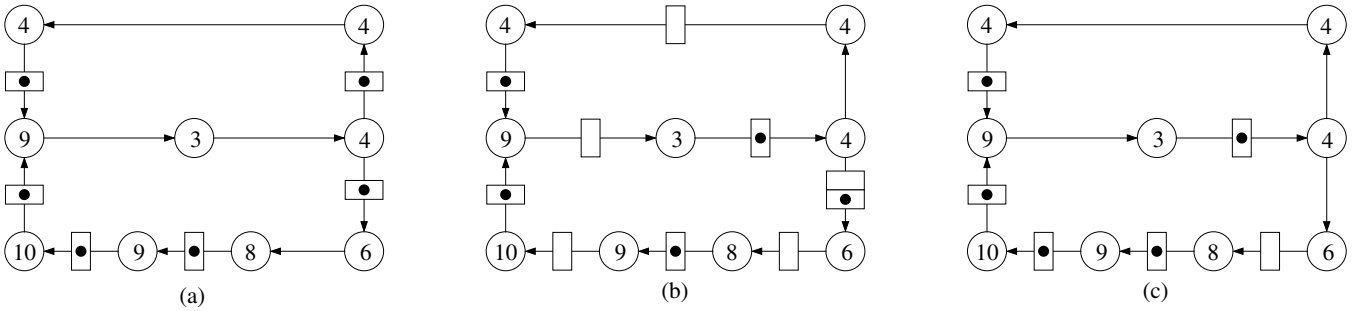

Fig. 1. (a) Retiming, (b) 2-slow retiming, (c) recycling and retiming.

units of the original design. In this case, the 2-slow solution can increase the frequency of the circuit at the expense of reducing the overall performance of the system. Note that every cycle in the circuit has doubled the number of registers in the original circuit.

Figure 1(c) shows an optimal configuration after combining retiming and recycling with a cycle time of 12 units. In this case, the throughput is determined by the bottom cycle in which a bubble has been inserted (4 dots in 5 registers). We can also say that the *processing rate*, i.e., the average number of cycles to process a dot, is now 5/4. This gives an *effective cycle time* of 15 time units ($15 = 12 \cdot 5/4$). It means that a dot is processed every 15 time units on average, compared to the 16 time units of the original design.

This examples illustrates the extra flexibility provided by R&R with regard to other performance optimization techniques.

E. Contributions

This paper presents an exact formulation of the R&R problem that can be specified with mixed ILP models. Even having a worst-case exponential complexity, the model can be practically solved for large systems.

The presented formulation can be seen as a general model for a variety of problems. *Retiming* is the particular case in which no bubbles are created. *Recycling* (without retiming) is reduced to finding a solution in which the original registers of the circuit are not moved, i.e., only bubbles are inserted to break long combinational cycles. Finally, the model also accepts an interpretation for the *c-slow* retiming problem.

The model presented in the paper resorts to the structural theory of marked graphs, which provides the support of linear algebraic techniques for the analysis of these systems. The analogy of the retiming problem with the reachability problem in marked graphs also brings a new result in the area, that is explained in the interpretation of theorem 2.2.

As expected, the combined formulation of the R&R problem provides a new space of solutions that cannot be explored when solving each of the above problems individually.

This paper is not dealing with the calculation of the initial state of the circuit. Techniques similar to those used for retiming [15] should be studied and adapted for this new type of problems.

The model presented in this paper assumes that back-pressure will never be constraining the performance of the system. This assumption can always be met by properly sizing the capacity of the registers inserted between the combinational blocks [16].

II. Marked graphs and retiming

Retiming has a strong analogy with the reachability problem in *marked graphs* (MG). In this section we will reformulate the retiming and R&R problems as reachability problems in MGs. This reformulation is not capricious. Some of the results in the theory of MGs can be reused in our context, thus providing an essential support to make new contributions in this area.

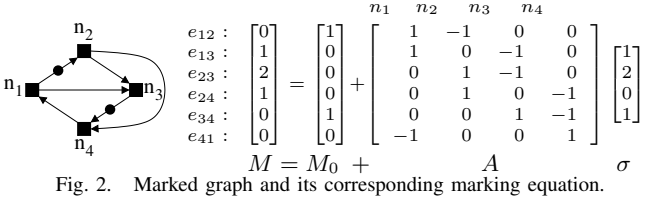

Fig. 2. Marked graph and its corresponding marking equation.

A. Marked graphs

Marked graphs are a special class of Petri nets [17] in which places have exactly one input and one output transition. The notation used in this paper is next presented.

Definition 2.1 (Marked graph (MG)): A *marked graph* is a triple (N, E, M_0), where N is a set of nodes, E is a set of edges and $M_0 : E \to \mathbb{N}$ is a marking that assigns an initial number of tokens to each arc. Given a node n, the notation ${}^\bullet n$ and $n^\bullet$ is used to denote the set of incoming and outgoing edges of n, respectively. Given a subset $\phi \subseteq E$, the total number of tokens of the edges in ϕ at a given marking M is denoted by $M(\phi)$. A node n is *enabled* at a marking M if $M(e) > 0$ for every $e \in {}^\bullet n$. An enabled node n can fire producing a new marking M' such that

$$M'(e) = \begin{cases} M(e) - 1 & \text{if } e \in {}^\bullet n \setminus n^\bullet \\ M(e) + 1 & \text{if } e \in n^\bullet \setminus {}^\bullet n \\ M(e) & \text{otherwise} \end{cases} \quad (1)$$

■

An example of MG is depicted in Fig. 2.

Retiming interpretation. The retiming graph of a circuit is isomorphic to a marked graph. Each combinational block corresponds to a node. Each connection corresponds to an edge. The registers in the retiming graph are represented by tokens in the MG. The firing rules of a MG coincide with the backward retiming rules: each time a node is retimed, registers are removed from the input edges and added to the output edges. ■

Without loss of generality, we will focus on *strongly connected marked graphs* (SCMG). In terms of retiming, a circuit can always be converted into strongly connected by adding a node X that represents the environment, and edges $X \to i$ and $o \to X$ for all inputs i and outputs o, respectively.

Definition 2.2 (Reachability and liveness): A marking M is called reachable if there is a firing sequence of transitions that leads from the initial marking M_0 to M. An MG is said to be live if every node can eventually fire from any reachable marking. ■

We next review two important properties of MGs and give their interpretation for the retiming problem.

Theorem 2.1 (Liveness [18]): An MG is *live* if every cycle ϕ is marked positively at M_0, i.e., $M(\phi) > 0$.
Retiming interpretation. Every cycle should have at least one register to avoid combinational cycles in the circuit netlist. ■

Next, a crucial result from MG theory that will be the basis of the ILP models presented in this paper.

Theorem 2.2 (Reachability [19]): A live marking M of an SCMG is reachable *iff* $M(\phi) = M_0(\phi)$ for every cycle ϕ.
Retiming interpretation. This property has two directions. The $\Rightarrow$ direction corresponds to a well-known result in retiming: a valid retiming preserves the number of registers at each cycle. The important direction is $\Leftarrow$ that provides a new result for the theory of retiming:

if an assignment of registers has the same number of registers at each cycle as the initial circuit, then the assignment is a valid retiming.

This result will be the basis of the ILP models proposed to solve the R&R problem. Implicitly, this duality also supports the fact that any solution obtained by retiming can also be obtained by a sequence of backward retiming steps. Thus, we can reduce the retiming problem to a reachability problem in MGs.

B. The marking equation

The *incidence matrix* A of an MG is an $|E|\times|N|$ matrix defined as follows. For every edge $e = (n_i, n_j)$: $A[e, n_i] = 1$, $A[e, n_j] = -1$ and $A[e, n_k] = 0$ for $k \neq i$ and $k \neq j$. A marking M is a $|E|$-vector that represents the number of tokens at every edge. A *firing vector* σ is an $|N|$-vector that represents the number of times each node is fired in a firing sequence. Another important theorem follows.

Theorem 2.3 ([19]): A nonnegative marking M is reachable *iff* the following *marking equation* holds:

$$M = M_0 + A\sigma \tag{2}$$

Retiming interpretation. M_0 is the initial assignment of registers to edges, M is the assignment after retiming, A is the retiming matrix and σ is the retiming vector.

An example on how to solve the marking equation is shown in Fig. 2. The marking equation is the basis for the well-known LP formulation of the retiming problem. The fact that A is totally unimodular guarantees a polynomial-time complexity even under the constraint that M must be integer.

C. Retiming and Recycling Graph

We now present the graph representation of an LI circuit.

Definition 2.3 (Retiming and Recycling Graph): A *Retiming and Recycling Graph* (RRG) is a four-tuple (N, E, R_0, δ), where (N, E, R_0) represents the underlying MG of the circuit and $\delta : N \rightarrow \mathbb{R}^+$ is a function that assigns a positive real number (delay) to every node. ■

R_0 represents an initial assignment of registers with informative data (dots) to the edges of the graph. Throughout the paper and without loss of generality, we will implicitly assume that all RRGs are strongly connected.

An important question is: what is a valid R&R configuration of a circuit? The answer is easy: let us take any valid retiming of the circuit and let us add any arbitrary number of registers (bubbles) to every edge. The new circuit is a valid R&R configuration. Therefore,

any integer solution for R and $\widehat{R}$ that fulfils the following expression:

$$R \geq \widehat{R} = R_0 + A\sigma \geq 0 \tag{3}$$

is a valid R&R solution. $\widehat{R}$ represents the *retiming subset* of the solution (the registers containing only dots; see theorem 2.3). R represents the R&R configuration (registers containing dots and bubbles). The difference between both vectors, $R - \widehat{R}$, represents the vector of registers containing the bubbles introduced by recycling.

III. Timing and performance

We now define some concepts related to timing and performance in RRGs.

Definition 3.1 (Combinational paths and cycle time): Given a register vector R of an RRG, a *combinational path* P is a sequence of edges $n_0 \xrightarrow{e_1} n_1 \xrightarrow{e_1} \cdots \xrightarrow{e_k} n_k$ such that $R(e_i) = 0$ for all edges in the path. The delay of the combinational path is

$$\delta(P) = \sum_{n_i \in P} \delta(n_i).$$

The cycle time associated to R, $\tau(R)$, is the maximum delay of all combinational paths. ■

Definition 3.2 (Cycle ratio and throughput): Given an RRG, a cycle C and a register vector R, its *cycle ratio* $\Theta_C(R)$ is defined as

$$\Theta_C(R) = \frac{\sum_{e\in C} R_0(e)}{\sum_{e\in C} R(e)}$$

The throughput $\Theta(R)$ achieved by a register assignment R is the minimum cycle ratio over all cycles in the RRG [20], i.e.,

$$\Theta(R) = \min_{\forall C} \Theta_C(R),$$

■

The throughput is the number of dots that can be processed by a node at each cycle. If a register assignment has no bubbles, then $\Theta(R) = 1$, as in Fig. 1(a). In Fig. 1(c) we have $\tau = 12$. The cycle ratios for the top and bottom cycles are 1 and 4/5, respectively. Therefore $\Theta(R) = 4/5$.

In general, retiming always produces solutions with $\Theta(R) = 1$, whereas recycling produces solutions with $\Theta(R) < 1$. In case of c-slow retiming, the solutions have $\Theta(R) = 1/c$. We now relate throughput and cycle time.

Definition 3.3 (Effective cycle time): Given a register assignment R, the *effective cycle time* is defined as

$$\mathbb{C}(R) = \tau(R)/\Theta(R).$$

The *effective cycle time* would be the cycle time of a conventional circuit that could process data at the same rate as the LI circuit. In the example of Fig. 1(c), $\mathbb{C} = 12 \cdot 5/4 = 15$. The inverse of $\mathbb{C}$ is what is known as *processing rate*. The main goal of this paper is to propose methods that minimize $\mathbb{C}$, i.e., maximize the processing rate.

Lemma 3.1 (Effective cycle time lower bound): Given an RRG and a register assignment R, then

$$\mathbb{C}(R) \geq \max_{c\in C} \frac{\sum_{v\in c} \delta(v)}{\sum_{e\in c} R_0(e)}, \tag{4}$$

where C is the set of all cycles of the RRG.

Proof: A known result from [21] indicates that for any cycle c

$$\tau(R) \geq \frac{\sum_{v \in c} \delta(v)}{\sum_{e \in c} R(e)}$$

We also know that

$$\Theta(R) \leq \Theta_c(R) = \frac{\sum_{e \in c} R_0(e)}{\sum_{e \in c} R(e)}$$

By using the previous inequalities,

$$\mathbb{C}(R) = \frac{\tau(R)}{\Theta(R)} \geq \frac{\sum_{v \in c} \delta(v)}{\sum_{e \in c} R_0(e)}$$

Therefore, inequality (4) holds. ∎

The lower bound for the effective cycle time coincides with the lower bound for *min-delay retiming* [21]. However, R&R offers more opportunities than retiming to approach this bound. For example, the lower bound on $\mathbb{C}$ in Fig. 1(a) is 12.25 time units. With min-period retiming, a cycle time of 16 can be achieved, while R&R can find a configuration with $\mathbb{C} = 15$. The impact of this bound on the potential benefits of R&R will be discussed in Section V-A.

A. Combinational path constraints

In order for an LI circuit to meet a cycle time τ, all combinational paths must be shorter than τ. In the retiming problem, these constraints are formulated by using the matrices W (minimum latency) and D (maximum delay) [1]. Unfortunately, this formulation is not valid for the R&R problem, since the number of registers between any pair of nodes can be changed by inserting any arbitrary number of bubbles. We next propose a different set of linear inequalities to formulate the combinational path constraints.

For every edge $e = (u, v)$ of the RRG, we define two variables, $t^{in}(e)$ and $t^{out}(e)$. The variable $t^{in}(e)$ represents the longest delay from any register to the entry of edge e, including the delay of the source node u. The variable $t^{out}(e)$ represents the longest delay from any register to the exit of edge e. If e has no registers, then $t^{out}(e) = t^{in}(e)$, otherwise $t^{out}(e) = 0$, since the exit of edge e represents the beginning of a combinational path. This definition of t^{in} and t^{out}, for every edge $e = (u, v)$, can be represented by the following constraints:

$$t^{in}(e) \geq t^{out}(e') + \delta(u) \qquad \forall e' = (w, u) \tag{5}$$
$$t^{out}(e) \geq t^{in}(e) - \tau^* R(e) \tag{6}$$
$$t^{out}(e) \geq 0, \quad t^{in}(e) \leq \tau \tag{7}$$

The constraint (5) indicates that the length of the path arriving at edge e is longer than the path exiting from a predecessor edge (e') plus the delay of the source node of e. The constraint (6) transfers the length of the path to the exit of the edge. In case the edge has some register ($R(e) > 0$), a new combinational path starts at edge e and $t^{out}(e) = 0$. τ^* is a constant large enough to guarantee its value to be larger than any possible value of τ. It is sufficient for τ^* to take the value of the cycle time in the original circuit. With τ^* being constant, all the inequalities are linear.

For the previous system of constraints, only the t^{in} variables are essential. The t^{out} variables can be eliminated by substitution.

Register delays. The previous constraints can be modified to account for the register delays in the combinational paths. If we call d_R the delay of a register, we can add d_R to the delay of each combinational path by assigning this delay to the beginning of the path. This is achieved by simply adding the constraint $t^{out}(e) \geq d_R$, for all edges of the graph.

Henceforth, the constraints (5-7) for a given register assignment R and cycle time τ will be represented by the predicate `Path_Constraints`(R, τ).

Lemma 3.2: Given an RRG with a register assignment R, $\tau(R) \leq \tau$ iff `Path_Constraints`(R, τ) is feasible.

Proof: $\Rightarrow$ Assume that $\tau(R) = \tau$. For each edge $e = (u, v) \in E$, we define $t^{in}(e)$ and $t^{out}(e)$ as follows:

$$t^{in}(e) = \delta(u) + \text{delay of the longest combinational path arriving at } u$$
$$t^{out}(e) = \begin{cases} t^{in} & \text{if } R(e) = 0 \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to check that the inequalities in (5-7) hold.

$\Leftarrow$ Assume that the set of constraints (5-7) has a feasible solution $(t^{in}_1, \ldots, t^{in}_m, t^{out}_1, \ldots, t^{out}_m)$ for some τ. Let $P = v_1, v_2, \ldots, v_{k-1}, v_k$ be a combinational path with delay $\delta(P)$. Then the following chain of inequalities holds:

$$t^{in}_{(v_1,v_2)} \geq \delta(v_1) \qquad \text{(constraint (5))}$$
$$t^{out}_{(v_1,v_2)} \geq t^{in}_{(v_1,v_2)} \geq \delta(v_1) \qquad \text{(constraint (6))}$$
$$t^{in}_{(v_2,v_3)} \geq t^{out}_{(v_1,v_2)} + \delta(v_2) \geq \delta(v_1) + \delta(v_2) \qquad \text{(constraint (5))}$$
$$\ldots$$
$$t^{in}_{(v_{k-1},v_k)} \geq \sum_{i=1}^{k-1} \delta(v_i)$$
$$t^{in}_{(v_k,v_j)} \geq \sum_{i=1}^{k} \delta(v_i) = \delta(P) \qquad \text{for any } v_j \text{ successor of } v_k$$

Since $t^{in}(e) \leq \tau$, for any e, we can conclude that $\delta(P) \leq \tau$ for any path P. ∎

B. Throughput constraints

We next present a constraint guaranteeing that a specific R&R configuration has at least throughput Θ.

The throughput constraint is as follows:

$$R \leq (R_0 + A\sigma)/\Theta \tag{8}$$

Lemma 3.3: Let R be a valid *R&R* register assignment. There is a nonnegative real vector σ that fulfils inequality (8) iff $\Theta(R) \geq \Theta$.

Proof: Let us call $R' = A\sigma$ and $R'(e)$ the component of R' associated to edge e. Therefore, the constraint (8) can be rewritten as $\Theta R \leq R_0 + R'$.

($\Rightarrow$) We will prove that every cycle C has a cycle ratio $\Theta_C(R) \geq \Theta$. Thus, using inequality (8) in the denominator:

$$\Theta_C(R) = \frac{\sum_{e \in C} R_0(e)}{\sum_{e \in C} R(e)} \geq \frac{\Theta \cdot \sum_{e \in C} R_0(e)}{\sum_{e \in C} (R_0(e) + R'(e))}$$

Since C is a cycle, the number of tokens in C remains invariant after any firing sequence (see theorem 2.2). Therefore,

$$\sum_{e \in C} R'(e) = 0$$

and consequently $\Theta_C(R) \geq \Theta$.

($\Leftarrow$) This part of the proof is easy but tedious. We give a sketch based on the results of previous authors. First of all, for results about throughput (dots per cycle), delays must be interpreted as unit delays coming from the registers (combinational logic has zero delay).

For a timed marked graph (N, E, M_0, δ) with incidence matrix B and throughput Θ, in which each node n has a delay $\delta(n)$, there exist an average marking (nonnegative real) and real vector σ', such that $\overline{M} = M_0 + B\sigma'$ and the following property holds for any $n \in N$ [22]:

$$\delta(n) \cdot \Theta \leq \overline{M}(e) \qquad \forall e \in {}^\bullet n. \tag{9}$$

In our case, the delays are associated to the registers on the edges (each register has unit delay). The previous result can be extended to marked graphs with delays on the edges[1]. From the average marking $\overline{M}$ it is easy to construct a nonnegative real marking $\widehat{R}$ such that $R \cdot \Theta \leq \widehat{R} = R_0 + A\sigma$, thus completing the proof. ∎

IV. ILP MODELS FOR RETIMING AND RECYCLING

This section presents ILP models for different variations of the R&R problem.

The main result that can be derived from the previous sections is the following:

Theorem 4.1: Given a cycle time τ and a throughput Θ, R is a valid *R&R* register assignment of an RRG (N, E, R_0, δ) with $\tau(R) \leq \tau$ and $\Theta(R) \geq \Theta$ iff there exists a feasible solution of the ILP

$$\texttt{RR}(\tau, \Theta) \equiv \begin{cases} R \geq R_0 + A\sigma_1 \geq 0 \\ R \leq (R_0 + A\sigma_2)/\Theta \\ \texttt{Path\_Constraints}(R, \tau) \\ R \text{ and } \sigma_1 \text{ are integer vectors.} \end{cases} \tag{10}$$

Proof: It immediately follows from expression (3) and lemmas 3.2 and 3.3. ∎

Note that the model does not include the vector $\widehat{R}$ from expression (3). To guarantee that $\widehat{R}$ is integer, the constraint of σ_1 being integer is included. After that, the value of $\widehat{R}$ is irrelevant to obtain a solution for R.

Henceforth, we will use $\texttt{RR}(\tau, \Theta)$ to denote the ILP model (10). This notation will also be used to denote a function that returns a register assignment, e.g., $R := \texttt{RR}(\tau, \Theta)$.

A. Min-period R&R

The min-period R&R problem can be formulated as follows:

> Given an RRG and a throughput $\Theta > 0$, find a register assignment R that minimizes the cycle time and has throughput $\Theta(R) \geq \Theta$.

From theorem 4.1, it immediately follows that R is the optimal solution of the following ILP, where Θ is a constant:

$$\texttt{MIN\_PER}(\Theta) \equiv \begin{cases} \min \tau \\ \text{subject to } \texttt{RR}(\tau, \Theta). \end{cases} \tag{11}$$

Similarly as before, $\texttt{MIN\_PER}(\Theta)$ will also denote a function that returns the register assignment R with minimum period.

[1]It is easy to see that each edge $u \to v$ can be transformed into two edges $u \to w \to v$. In this way, the delay of the edge can be associated to the node w, thus having a marked graph with delays on the transitions only.

B. Max-throughput R&R

The formulation of the problem is as follows:

> Given an RRG and a cycle period τ, find a register assignment R with $\tau(R) \leq \tau$ that maximizes the throughput $\Theta(R)$.

The problem can also be solved by using a model derived from theorem 4.1, in which τ is a constant:

$$\texttt{MAX\_THR\_NL}(\tau) \equiv \begin{cases} \max \Theta \\ \text{subject to } \texttt{RR}(\tau, \Theta). \end{cases} \tag{12}$$

However, the model $\texttt{MAX\_THR\_NL}(\tau)$ is not linear since Θ is now a variable of the model and the second constraint of $\texttt{RR}(\tau, \Theta)$ is not linear. Unfortunately, the model is not convex either, thus discarding the possibility of using a solver for convex problems.

1) Binary search for max-throughput: An option to solve the previous problem is to perform a search on the possible values of Θ. With this approach, the model becomes linear when Θ is a constant value. In this paper we propose to perform a binary search for Θ over the interval $(0, 1]$.

A binary search always explores values of Θ in an interval $[\Theta_L, \Theta_U]$ where a feasible solution exists for Θ_L but not for Θ_U. An important question to answer is the following: what is the size of the interval that guarantees Θ_L to be the optimal solution?

Theorem 4.2: Let $\Theta_L = n_L/m_L$ and $\Theta_U = n_U/m_U$ be two possible values of the throughput for an RRG such that $\Theta_L \neq \Theta_U$. Then,

$$|\Theta_L - \Theta_U| \geq \frac{1}{(|R_0| + |E|)^2}$$

where $|R_0|$ represents the number of initial registers and $|E|$ is the number of edges in the graph.

Proof: Let $\frac{n_1}{m_1}$ and $\frac{n_2}{m_2}$ be two different rational numbers with denominators not larger than m. Then

$$\left| \frac{n_1}{m_1} - \frac{n_2}{m_2} \right| = \left| \frac{n_1 m_2 - n_2 m_1}{m_1 m_2} \right| \geq \frac{1}{m^2}.$$

We also know that the maximum number of registers we can have in a cycle for a configuration with maximum throughput is not larger than the number of initial registers plus the number of edges (bubbles than can be inserted in the graph)[2]. The proof of the theorem immediately follows from this observation. ∎

A tighter bound could be calculated by finding the longest simple cycle in the graph. However, this problem is NP-complete [23].

2) Min-area max-throughput R&R: Since the binary search for Θ converts the model (12) into a linear form, the cost function can be customized to minimize area (number of registers) in the following way:

$$\texttt{MAX\_THR}(\tau) \equiv \begin{cases} \min \sum_{e \in E} R(e) \\ \text{subject to } \texttt{RR}(\tau, \Theta) \\ \text{Binary search on } \Theta. \end{cases} \tag{13}$$

C. Minimum effective cycle time

The formulation of the problem is the following:

> Given an RRG, find a register assignment $R_{\min}$ such that the effective cycle time $\mathbb{C}(R_{\min})$ is minimized.

[2]Putting more than one bubble in an edge degrades the throughput without improving the cycle period. For this reason, only configurations with at most one bubble per edge must be considered.

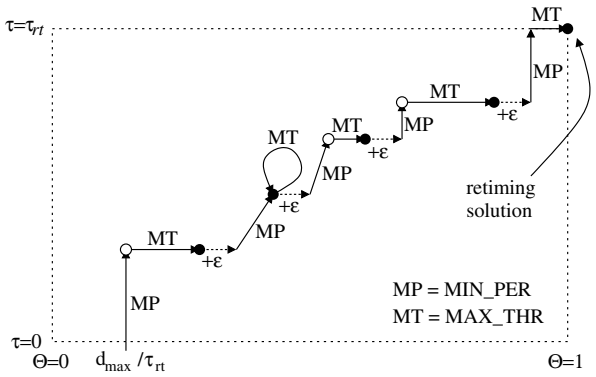

Fig. 3. Search for minimum effective cycle time.

We will show how the ILP models MIN_PER and MAX_THR can be combined to find $R_{\min}$. We first start by a preliminary result required to make the search efficient.

Theorem 4.3: $\Theta(R_{\min}) \geq \frac{d_{\max}}{\tau_{rt}}$, where $d_{\max}$ is the maximum delay of a node and τ_{rt} is the cycle period obtained by min-delay retiming.

Proof: Every retiming configuration is also a valid R&R assignment. Therefore $\mathbb{C}(R_{\min}) \leq \tau_{rt}$. By the definition of $\mathbb{C}$ (see Def. 3.3), and using the fact that $\tau(R_{\min}) \geq d_{\max}$,

$$\Theta(R_{\min}) = \frac{\tau(R_{\min})}{\mathbb{C}(R_{\min})} \geq \frac{d_{\max}}{\tau_{rt}}$$

■

Then search for the optimal $\mathbb{C}$ can be performed by interleaving MIN_PER and MAX_THR.

```
MIN_EFF_CYC_STEP(Θ):
    R1 := MIN_PER(Θ);
    R2 := MAX_THR(τ(R1));
    return R2;
```

The intuition behind this strategy is the following. Given a target Θ, a register assignment R_1 with minimum period is obtained by MIN_PER(Θ). R_1 has a throughput not smaller than Θ. After that, another register assignment R_2 maximizing the throughput is obtained by MAX_THR($\tau(R_1)$). R_2 is guaranteed to have a clock period not larger than $\tau(R_1)$.

This process can be iteratively executed until the retiming solution is found ($\Theta = 1$, $\tau = \tau_{rt}$). For every explored solution R, the effective cycle time $\mathbb{C}(R)$ can be calculated. At the end of the process, the solution with minimum $\mathbb{C}$ is returned.

Formally, the procedure to find $\mathbb{C}(R_{\min})$ is described by the following algorithm:

```
MIN_EFF_CYC(RRG):
    C := τrt;                          {τ from min-period retiming}
    Θ := dmax/τrt;                     {from theorem 4.3}
    ε := 1/(|R0|+|E|)^2;               {from theorem 4.2}
    while (Θ < 1)
        R := MIN_EFF_CYC_STEP(Θ);
        if (C(R) < C) then C := C(R);
        Θ := Θ(R) + ε;
    return C;
```

Experimental results show that this search never makes more then 10 iterations for graphs with up to 1000 edges.

Figure 3 illustrates the search for the optimum $\mathbb{C}$ through a diagram that represents Θ in the x-axis and τ in the y-axis. The MP and MT labels indicate the progress performed by MIN_PER and MAX_THR, respectively. The search teminates when the retiming solution is found. In some cases, MAX_THR does not make any progress (e.g., see the loop with label MT in the diagram). In those cases, the ε increase guarantees the termination. The next theorem guarantees that the search does not miss any solution that could provide a better effective cycle time.

Theorem 4.4: Let R_1 be a solution of MIN_PER(Θ) and R_2 the register assignment obtained by MAX_THR($\tau(R_1)$). Let R' be any other arbitrary register assignment such that $\Theta \leq \Theta(R') \leq \Theta(R_2)$. Then, $\mathbb{C}(R') \geq \mathbb{C}(R_2)$.

Proof: By contradiction. Let us assume that R' satisfies the conditions of the theorem and $\mathbb{C}(R') < \mathbb{C}(R_2)$, or equivalently $\frac{\tau(R')}{\Theta(R')} < \frac{\tau(R_2)}{\Theta(R_2)}$.

Since $\Theta(R') < \Theta(R_2)$ it follows that $\tau(R') < \tau(R_2)$. But R_2 is the solution of the MAX_THR($\tau(R_1)$) and hence $\tau(R_2) \leq \tau(R_1)$. Therefore, $\tau(R') < \tau(R_1)$, but this contradicts the fact that MIN_PER(Θ) produces R_1, which has the minimum cycle time from all the assignments with throughput at least Θ. ■

Therefore, the register assignments obtained by solving MAX_THR are the only ones that must be considered during the search. They correspond to the black circles in Fig. 3.

D. C-slow retiming

For completeness, a mode for c-slow retiming is presented.

$$\text{C\_SLOW}(\tau) \equiv \begin{cases} \min \alpha c + \sum_{e \in E} R(e) \\ R = cR_0 + A\sigma \geq 0 \\ \text{Path\_Constraints}(R, \tau) \\ \sigma \text{ and } c \text{ are integer} \end{cases} \quad (14)$$

The new constraint is $R = cR_0 + A\sigma$, which substitutes the initial register assignment R_0 by cR_0, indicating that each register in the original circuit is substituted by c registers (one dot and $c-1$ bubbles).

The cost function minimizes c assuming that α is a large constant. After that, the register count is minimized.

V. Experimental results

The first consideration for R&R is that it will mostly be applied to coarse levels of granularity, e.g., at the level of 16-, 32- or 64-bit registers in medium and large systems having few dozens or hundreds of computational blocks. Unfortunately, there is no set of benchmarks usable by academia that keep hierarchical information and can be effectively used for realistic experiments. For this reason, we designed synthetic experiments based on the underlying graphs of the sequential ISCAS circuits.

For every circuit, the following transformations were performed:

- The original latches were removed, and every gate was considered to be a large combinational block. Edges were considered to be n-bit channels. Each combinational block was assigned a delay generated randomly from a uniform distribution in the interval $(0, 20]$.
- The largest strongly connected component of the graph was kept in the graph. The rest of nodes and edges were removed[3].

[3]An alternative way to ensure strongly connectedness would have been to connect the inputs and the outputs with a fake node. However, we found this approach unrealistic due to the fact that an artificially congested node appeared for each example.

TABLE I
EXPERIMENTAL RESULTS.

	Initial					Retiming		Recycling ($\tau \leq 0.75\tau_{rt}$)				Retiming+Recycling ($\tau \leq 0.75\tau_{rt}$)				
	$\|V\|$	$\|E\|$	R_0	τ	$\mathbb{C}^*$	R	τ_{rt}	R	τ	Θ	$\mathbb{C}$	R	τ	Θ	$\mathbb{C}$	CPU
s27	14	24	14	58.20	58.20	5	58.20	12	30.32	0.50	60.64	12	30.32	0.50	60.64	00:00:01
s208	8	9	2	87.58	87.58	2	87.58	3	51.00	0.50	102.00	3	49.86	0.50	99.72	00:00:01
s344	135	176	76	72.52	39.75	72	41.53	106	26.15	0.50	52.30	101	23.90	0.50	47.80	00:00:16
s349	135	176	76	134.70	46.25	78	48.64	114	27.46	0.50	54.92	95	27.46	0.50	54.92	00:12:00
s382	42	60	23	66.00	38.14	29	41.32	41	24.39	0.50	48.78	34	24.39	0.50	48.78	00:00:02
s386	48	131	64	49.30	43.75	40	46.02	84	25.12	0.50	50.24	76	25.12	0.50	50.24	00:00:04
s400	46	66	29	66.24	50.42	34	52.43	42	33.24	0.50	66.48	39	24.88	0.43	58.05	00:21:26
s420	8	9	2	76.70	76.70	2	76.70	3	40.59	0.50	81.18	3	40.59	0.50	81.18	00:00:01
s444	58	82	41	80.47	52.69	35	55.02	44	31.37	0.50	62.74	38	39.73	0.67	59.60	00:00:09
s510	103	407	197	69.72	64.48	36	64.48	145	34.65	0.50	69.30	132	34.65	0.50	69.30	00:26:27
s526	50	71	31	80.66	80.66	23	80.66	29	45.38	0.50	90.76	30	41.44	0.50	82.88	00:00:01
s641	221	270	133	123.09	40.85	117	43.97	178	25.83	0.50	51.66	137	30.90	0.67	46.35	00:05:19
s713	256	341	158	171.65	40.21	143	49.93	252	23.57	0.42	56.57	240	36.72	0.75	48.96	00:18:25
s820	110	424	205	53.61	53.23	283	53.46	288	31.04	0.50	62.08	122	31.04	0.50	62.08	00:10:09
s832	117	462	226	61.84	49.04	329	50.39	354	30.45	0.50	60.90	231	27.40	0.50	54.80	00:21:42
s838	8	9	2	68.40	68.40	2	68.40	4	38.33	0.50	76.66	3	35.03	0.50	70.06	00:00:01
s953	268	371	172	90.93	50.69	131	57.47	234	25.99	0.40	64.98	158	35.88	0.67	53.82	01:29:40
s1488	133	572	284	72.52	58.05	123	63.39	194	37.05	0.50	74.10	196	36.01	0.50	72.02	00:03:58
s1494	136	572	275	71.10	58.22	306	60.29	393	33.13	0.50	66.26	231	33.13	0.50	66.26	01:48:08

- Each channel was assigned an n-bit register with a certain probability. The probability was chosen to be 0.5. Thus, about half of the channels were assigned a register, whereas the other half were just wires.

For each example, several solutions were obtained using the optimization techniques presented in the paper. After satisfying the performance constraints of the model, all solutions were optimized for minimum register count.

Table I reports the experimental results. The "Initial" columns report the parameters of the example. The column $\mathbb{C}^*$ shows the lower bound on the effective cycle time (see lemma 3.1). The results obtained by min-period retiming are reported in the next two columns.

The configurations for "Recycling" and "Retiming and recycling" were obtained by imposing the constraint $\tau \leq 0.75\tau_{rt}$, i.e., the target cycle period was required to be smaller than 3/4 of the one obtained by retiming. As an example, the target cycle period for `s27` was defined to be $\tau \leq 0.75 \cdot 58.20 = 43.65$. The obtained results had the optimum effective cycle time ($\mathbb{C}$) satisfying the cycle period constraint.

The "Recycling" solution was obtained by only inserting bubbles in the circuit and not touching the registers defined by the min-period retiming[4]. You can observe that the period is often smaller than the target period $0.75\tau_{rt}$. The increase in number of registers is strictly associated to the bubbles inserted by recycling.

The R&R solutions ("Retiming+recycling") were obtained by `MIN_EFF_CYC`, as explained in Section IV-C. The results show the benefits in performance and area (register count) when combining retiming and recycling in the same model. In some cases, the reduction of effective cycle time is significant e.g., `s526`, `s641` and `s953`. In other cases, the performance improvement is not so relevant but the reduction in register count is important, e.g., `s820`, `s832` and `s1494`. An interesting case is `s400`, in which the throughput of the R&R solution (0.43) is smaller than the one of the recycling solution (0.50). However, the reduction in cycle time (from 33.24 to 24.88) results in a superior performance. This is a clear example of the diversity of solutions that can be explored by R&R.

The CPU time (hh:mm:ss) is only reported for the R&R solutions, which corresponds to the most complex optimization model. CPLEX [24] was used as ILP solver. To make the computations affordable, a timeout of 600 seconds was defined for each ILP model. It is important to recall that `MIN_EFF_CYC` generates several ILP models for each search, coming from the *while* loop of the algorithm and the binary search performed by `MAX_THR`. Even with this limitation of CPU time, the results show that significant improvements in performance can still be obtained, even without guaranteeing optimality.

Results from 2-slow retiming were also obtained, for completeness. All of them were worse than the R&R configurations. Given their irrelevance for the main conclusions of the paper, they are not shown in the table.

[4]This was achieved by adding constraints that fix the location of dots in ILP $\texttt{RR}(\tau, \Theta)$.

A. Minimum effective cycle time

The same experiments were run without imposing the constraint $\tau \leq 0.75\tau_{rt}$ on the cycle period, thus obtaining the optimum effective cycle time. As it was expected, the optimum effective cycle time was achieved by min-period retiming ($C = \tau_{rt}$) in almost all cases. Only in two cases, `s713` and `s953`, the insertion of bubbles contributed to slightly improve the performance. In these two cases, the solutions coincided with the ones obtained with the $\tau \leq 0.75\tau_{rt}$ constraint, and reported in Table I.

The reason for these results is easy to explain. The only potential margin for improvement is the difference $\tau_{rt} - \mathbb{C}^*$ (see Table I), which was usually small in our examples. Even with the existence of the margin, the degradation in throughput produced by the insertion of bubbles reduces the chances for improvement significantly. The two successful cases are the ones in which the difference $\tau_{rt} - \mathbb{C}^*$ is larger.

The potential performance improvements of R&R requires a more profound study in the future. A simple study for rings was presented in [14]. However, it is difficult to extend those conclusions for more complex circuits. A preliminary intuition indicates that only graphs with long cycles and unbalanced delays can increase the chances for optimization.

VI. CONCLUSIONS

A general ILP model for retiming and recycling has been presented. It covers previous models for sequential logic synthesis and allows to combine them in a collaborative way to explore a new space of solutions.

This approach opens the door to new optimization engines targetting at delay, throughput or area. The recent approaches for latency-insensitive design that reduce data and control overhead may provide efficient implementations of sequential circuits that can contribute to improve the performance of nanoelectronic systems.

Acknowledgements. This research has been funded by a grant from Intel Corp., CICYT TIN 2004-07925, the FPU grant AP2005-4866, a Distinction for the Promotion of Research by the Generalitat de Catalunya and a grant from AGAUR (2005PIV1-59).

REFERENCES

[1] C. E. Leiserson and J. B. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, no. 1, pp. 5–35, 1991.

[2] N. V. Shenoy, "Retiming: Theory and practice." *Integration, the VLSI Journal*, vol. 22, no. 1, pp. 1–21, 1997.

[3] S. S. Sapatnekar and R. B. Deokar, "Utilizing the timing skew equivalence in a practical algorithm for retiming large circuits," *IEEE Transactions on Computer-Aided Design*, vol. 15, no. 10, pp. 1237–1248, Oct. 1996.

[4] C. Lin and H. Zhou, "Retiming for wire pipelining in systems-on-chip," in *Proc. International Conf. Computer-Aided Design (ICCAD)*, Nov. 2003, pp. 215–220.

[5] V. Nookala and S. S. Sapatnekar, "A method for correcting the functionality of a wire-pipelined circuit," in *Proc. ACM/IEEE Design Automation Conference*, June 2004, pp. 570–575.

[6] L. P. Carloni, K. L. McMillan, A. Saldanha, and A. L. Sagiovanni-Vincentelli, "A methodology for correct-by-construction latency insensitive design," in *Proc. International Conf. Computer-Aided Design (ICCAD)*, Nov. 1999, pp. 309–315.

[7] L. P. Carloni and A. L. Sangiovanni-Vincentelli, "Performance analysis and optimization of latency insensitive systems," in *Proc. ACM/IEEE Design Automation Conference*, June 2000, pp. 361–367.

[8] T. E. Williams, "Performance of iterative computation in self-timed rings," *Journal of VLSI Signal Processing*, vol. 7, no. 1/2, pp. 17–31, Feb. 1994.

[9] R. Manohar and A. J. Martin, "Slack elasticity in concurrent computing," in *Proc. 4th International Conference on the Mathematics of Program Construction*, ser. Lecture Notes in Computer Science, J. Jeuring, Ed., vol. 1422, 1998, pp. 272–285.

[10] P. A. Beerel, N.-H. Kim, A. Lines, and M. Davies, "Slack matching asynchronous designs," in *Proc. of the 12th Int. Symp. on Asynchronous Circuits and Systems*, 2006.

[11] P. Prakash and A. J. Martin, "Slack matching quasi delay-insensitive circuits," in *Proc. of the 12th Int. Symp. on Asynchronous Circuits and Systems*, 2006.

[12] M. R. Casu and L. Macchiarulo, "A new approach to latency insensitive design," in *Proc. Digital Automation Conference (DAC)*, June 2004, pp. 576–581.

[13] J. Cortadella, M. Kishinevsky, and B. Grundmann, "Synthesis of synchronous elastic architectures," in *Proc. ACM/IEEE Design Automation Conference*, July 2006, pp. 657–662.

[14] L. P. Carloni and A. L. Sangiovanni-Vincentelli, "Combining retiming and recycling to optimize the performance of synchronous circuits," in *16th Symp. on Integrated Circuits and System Design (SBCCI)*, Sept. 2003, pp. 47–52.

[15] H. J. Touati and R. K. Brayton, "Computing the initial states of retimed circuits." *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 12, no. 1, pp. 157–162, 1993.

[16] R. Lu and C.-K. Koh, "Performance analysis of latency-insensitive systems," *IEEE Transactions on Computer-Aided Design*, vol. 25, no. 3, pp. 469–483, 2006.

[17] T. Murata, "Petri Nets: Properties, analysis and applications," *Proceedings of the IEEE*, pp. 541–580, Apr. 1989.

[18] F. Commoner, A. W. Holt, S. Even, and A. Pnueli, "Marked directed graphs," *Journal of Computer and System Sciences*, vol. 5, pp. 511–523, 1971.

[19] T. Murata, "Circuit theoretic analysis and synthesis of marked graphs," *IEEE Trans. Circuits and Systems*, vol. CAS-24, no. 7, pp. 400–405, July 1977.

[20] C. Ramchandani, "Analysis of asynchronous concurrent systems by Petri nets," Project MAC, TR-120, M.I.T., Cambridge, MA, 1974.

[21] M. C. Papaefthymiou, "Understanding retiming through maximum average-delay cycles," *Mathematical Systems Theory*, vol. 27, no. 1, pp. 65–84, 1994.

[22] J. Campos, G. Chiola, and M. Silva, "Ergodicity and throughput bounds for petri nets with unique consistent firing count vector," *IEEE Transactions on Software Engineering*, vol. 17, no. 2, pp. 117–125, 1991.

[23] C. Papadimitriou, *Computational Complexity*. Addison-Wesley, 1995.

[24] "CPLEX," `http://www.ilog.com/products/cplex`.